CMPG-767 Image Processing and Analysis

SPATIAL DOMAIN FILTERING. NOISE MODELS. OPTIMAL LINEAR FILTER

Linear Spatial Domain Filtering

- Linear spatial domain filtering of an image of size M x N with a filter of size *m* x *n* is defined by the expression

$$\hat{f}(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, g(x+s, y+t)$$

where $g(x, y)$ is the image to be processed,
a=[(m-1)/2], b=[(n-1)/2], and $w(s, t)$ form the filter kernel (mask)

Linear Spatial Domain Filtering

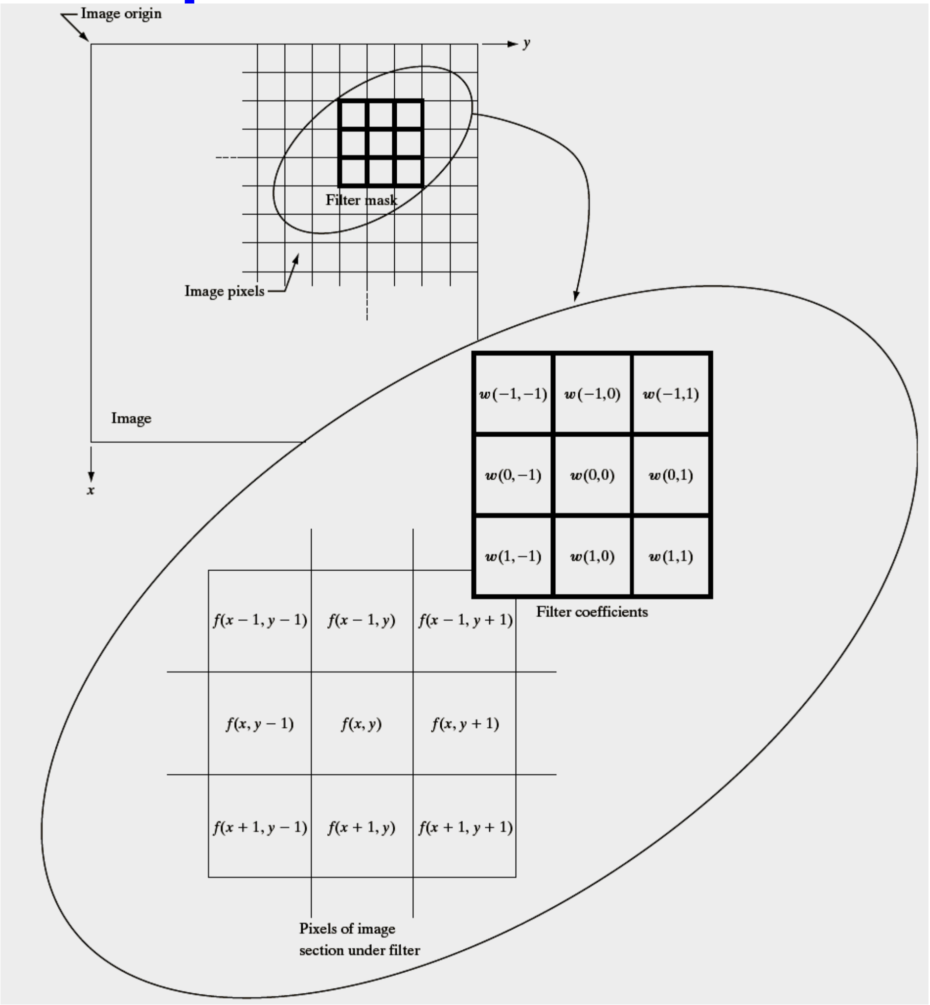
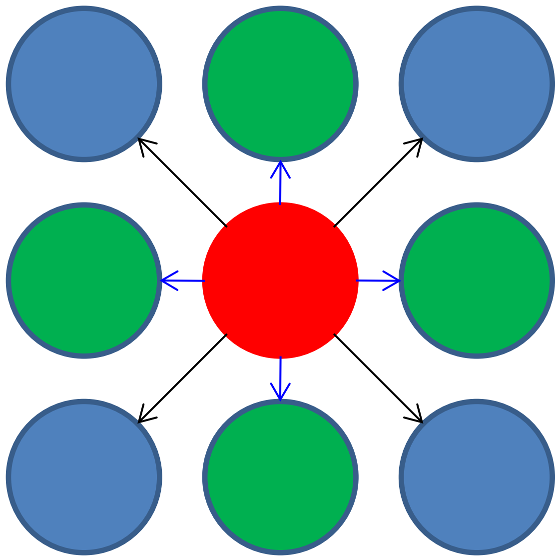


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

Smart (Gaussian-like) Filter Kernel

- Vertical and horizontal adjacent pixels (the green ones in the picture) in a 3x3 window are located "closer" to the center of the window than the diagonal pixels. Evidently, the distance from a center of a square to any of its sides is shorter than the one to its vertices:



Smart (Gaussian-like) Filter Kernel

- So it can be reasonable to emphasize contribution of the central pixel and its closest neighbors to the resulting intensity value. This can be done by the following filter mask (kernel)

$$\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Smart Filtering Kernel in General Case

- More smart filters can be designed from the following heuristic idea

$$\frac{1}{N}\begin{pmatrix} w_{-1,-1} & w_{-1,0} & w_{-1,1} \\ w_{0,-1} & w_{0,0} & w_{0,1} \\ w_{1,-1} & w_{1,0} & w_{1,1} \end{pmatrix}$$

$$w_{0,0} >> w_{-1,-1}, w_{-1,0}, w_{-1,1}, w_{0,-1}, w_{0,1}, w_{1,-1}, w_{1,0}, w_{1,1}$$

$$w_{-1,0}, w_{0,-1}, w_{0,1}, w_{1,0} > w_{-1,-1}, w_{-1,1}, w_{1,-1}, w_{1,1}$$

$$\sum_{i=-1}^{1}\sum_{j=-1}^{1} w_{i,j} = N$$

Smart Filtering Kernel in General Case

- Smart filtering with a 3x3 kernel may help to reduce smoothing edges and preserve small 1.5 x 1.5 details

$$\frac{1}{N}\begin{pmatrix} w_{-1,-1} & w_{-1,0} & w_{-1,1} \\ w_{0,-1} & w_{0,0} & w_{0,1} \\ w_{1,-1} & w_{1,0} & w_{1,1} \end{pmatrix}$$

$$w_{0,0} >> w_{-1,-1}, w_{-1,0}, w_{-1,1}, w_{0,-1}, w_{0,1}, w_{1,-1}, w_{1,0}, w_{1,1}$$

$$w_{-1,0}, w_{0,-1}, w_{0,1}, w_{1,0} > w_{-1,-1}, w_{-1,1}, w_{1,-1}, w_{1,1}$$

$$\sum_{i=-1}^{1}\sum_{j=-1}^{1} w_{i,j} = N$$

Optimal Spatial Domain Linear Filtering

- Does the best universal filtering mask exist?
- Since our criterion of the filtering quality is a square error, an optimal filtering kernel can be obtained by solving the following optimization problem: minimization of the expectation of the functional of a square error with respect to the weights:

$$E\left(f(x,y)-\hat{f}(x,y)\right)^2 =$$

$$= E\left(f(x,y)-\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\,g(x+s,y+t)\right)^2 = \min_{w(s,t)}$$

Optimal Spatial Domain Linear Filtering

- Let us differentiate the expectation of the error function with respect to the weight $w(k,l)$

$$\left[E\left(f(x,y)-\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)g(x+s,y+t)\right)^2\right]'=$$

$$2E\left(f(x,y)\cdot g(x+k,y+l)\right)-2\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)E\left(g(x+s,y+t)\cdot g(x+k,y+l)\right)$$

- Setting this derivative equal to 0 (a point of a minimum of the error function), we obtain the equation

$$E\left(f(x,y)\cdot g(x+k,y+l)\right)-\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)E\left(g(x+s,y+t)\cdot g(x+k,y+l)\right)=0$$

Optimal Spatial Domain Linear Filtering

$$E\left(f(x,y)\cdot g(x+k,y+l)\right)-\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)E\left(g(x+s,y+t)\cdot g(x+k,y+l)\right)=0$$

- Let us use the following notation

$$B_{fg}(k,l)=E\left(f(x,y)\cdot g(x+k,y+l)\right)$$

$$B_{gg}(k+s,l+t)=E\left(g(x+s,y+t)\cdot g(x+k,y+l)\right)$$

- $B_{fg}(k,l)$ is the **correlation** and $B_{gg}(k+s,l+t)$ is the **autocorrelation** at the pixel (k, l)
- https://en.wikipedia.org/wiki/Digital_image_correlation_and_tracking

$$B_{fg}(i,j)=\frac{\sum_m\sum_n[f(m+i,n+j)-\bar{f}][g(m,n)-\bar{g}]}{\sqrt{\sum_m\sum_n[f(m,n)-\bar{f}]^2\sum_m\sum_n[g(m,n)-\bar{g}]^2}}$$

Optimal Spatial Domain Linear Filtering

- Our equation is transformed as follows

$$B_{fg}(k,l)-\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)B_{gg}(k+s,l+t)=0$$

- Repeating the same for all $k=-a,\dots,a;l=-b,\dots,b$, we obtain the following system of nm linear algebraic equations with regard to (a Wiener-Hopf system) $w(s,t)$

$$B_{fg}(k,l)-\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)B_{gg}(k+s,l+t)=0;$$
$$k=-a,\dots,a;l=-b,\dots,b$$

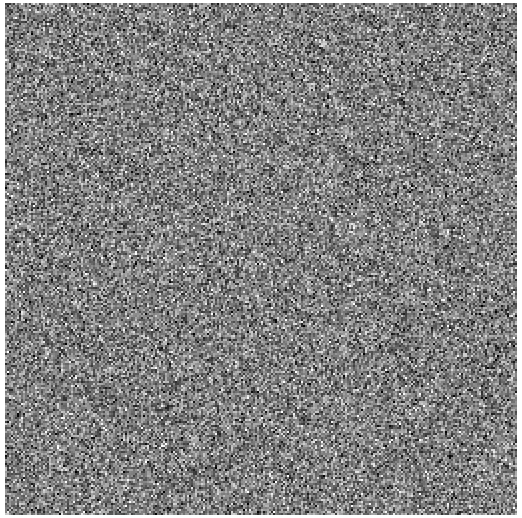
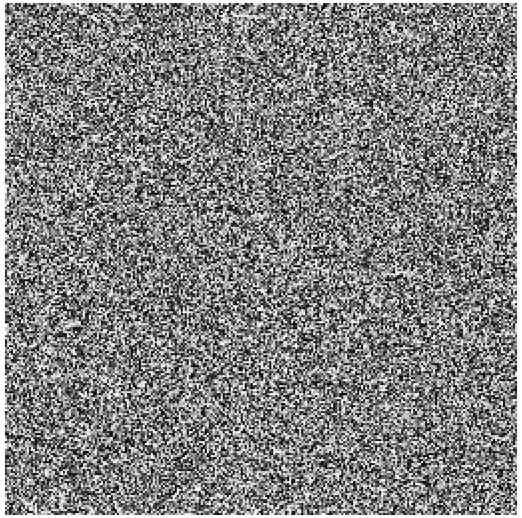
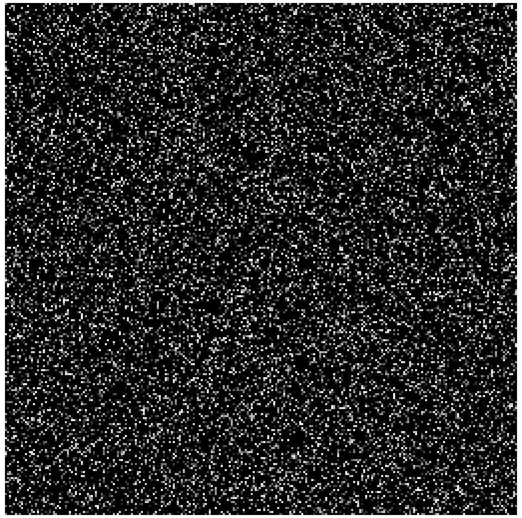
Optimal Spatial Domain Linear Filtering

$$B_{fg}(k,l) - \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) B_g(k+s, l+t) = 0;$$

$$k = -a, \dots, a; l = -b, \dots, b$$

- Everything should be great, but we do not know(!) $f(x,y)$ and, respectively, we cannot find $B_{fg}(k,l)$; we can just estimate it by the product of horizontal and vertical correlations of the initial noisy image $g(x,y)$
- Then the system of equations can easily be solved, and its solution gives the optimal (quasi-optimal, considering that $B_{fg}(k,l)$ is not exact, it is only estimated with some accuracy) linear filter kernel (mask) $w(s,t)$

Noise Models

Modeling of Noise Probability Density Functions

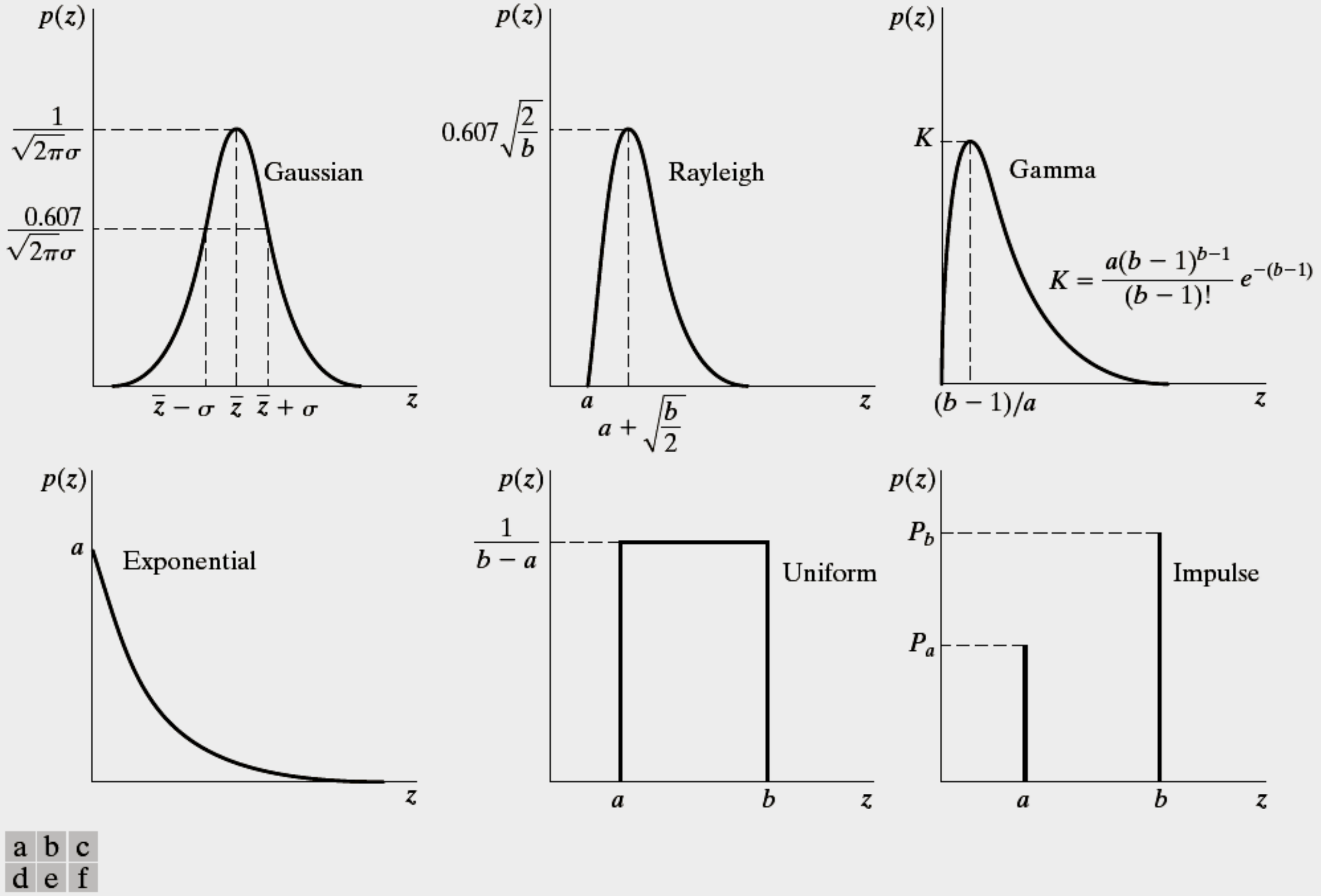


FIGURE 5.2 Some important probability density functions.

Modeling of Noise Probability Density Functions

- The most frequently appeared kinds of noise are Gaussian and impulse
- For example, even a single bit inversion in a communication channel immediately leads to creation of an impulse corrupting an image
- CCD sensors always generate additive Gaussian noise whose intensity depends particularly on the temperature and light conditions
- Images created by radars (radio and ultrasound) always suffer from the multiplicative (speckle) Gaussian noise

Gaussian Noise

- Gaussian noise is the most frequently used model of noise. It has the following probability density function

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}}$$

where z is the intensity value, $\bar{z}$ is the average intensity value, σ is the noise standard deviation

Gaussian Noise: Simulation

- To generate Gaussian noise in Matlab, we may use the **randn** function. To generate in using any high-level language, we may use for example, the Box-Muller algorithm
- Let r and φ are independent random variables with mean=0.5 and uniformly distributed in the interval [0,1]. Then

$$z_1 = \cos(2\pi\varphi)\sqrt{-2\ln r}$$

$$z_2 = \sin(2\pi\varphi)\sqrt{-2\ln r}$$

are normally distributed random variables on the interval [-1,1] with 0 mean and variance=1

Gaussian Noise: Simulation

- Box-Muller algorithm description
- m = mean; // the mean we need
  sigma = std; //the standard deviation we need
  r = rand(1);
  fi = rand(2);
  z1 = sqrt(-2 * Log[r]) * Sin[2 *pi * fi];
  z2 = sqrt(-2 * Log[r]) * Cos[2 * pi * fi];

  // normally distributed variables with given mean (m) and standard deviation (sigma)
  x1 = m + z1 * sigma;
  x2 = m + z2 * sigma;

Additive Gaussian Noise

- To simulate an additive Gaussian noise, it is necessary to generate a Gaussian random field having the same mean as a global mean of an image, to which the noise will be added (or with which it will be multiplied) and a given variance (usually, noise variance is measured in terms of the image variance (standard deviation): if σ is the image standard deviation, then noise variance can be 0.1σ, 0.2σ, 0.3σ, … (usually a noise heavier than 0.5σ is not considered)

Gaussian Noise: Simulation

- Then the generated noise $\eta(x, y)$ shall be added to the image, and the mean must be subtracted, to be sure that a noisy image has the same mean as a clean image and noise
- Additive noise

$$g(x, y) = f(x, y) + \eta(x, y) - m$$

where *m* is the mean of both image and noise

White Noise

- Additive noise with zero mean is called a white noise if it has equal power within a fixed bandwidth at any frequency.

- In other words, white noise equally contributes to a power corresponding to each frequency contained in a signal.

Impulse Noise

- Impulse noise may corrupt any signal including digital images just due to occasional inversion of a single bit representing the intensity value in some pixel
- The general model of impulse noise is

$$g(x,y)=\begin{cases} p_n, \eta(x,y) \\ 1-p_n, f(x,y) \end{cases}$$

where p_n is the probability of distortion (p_n in percents is called the corruption rate) $p_n \cdot 100\%$

Impulse Noise

- Unlike additive noise, which just distorts intensity values, impulse noise completely replaces the intensity values in those pixels that are corrupted.
- The higher is corruption rate, the more pixels are affected by noise and the more difficult is filtering

Salt-and-Pepper Impulse Noise

- Salt-and-Pepper impulse noise replaces the intensity values in the image $f(x,y)$ by 0s and 255s with some certain probabilities

$$g(x,y)=\begin{cases} p_0, & 0 \\ p_{255}, & 255 \\ 1-(p_0+p_{255}), & f(x,y) \end{cases}$$

- Since 0 is black and 255 is white, a corrupted image is covered by white and black impulses ("salt-and-pepper")
- The corruption rate is $(p_0+p_{255})\cdot 100\%$

Bipolar Impulse Noise

- Bipolar impulse noise replaces the intensity values in the image $f(x,y)$ by two values n_1 and n_2 with some certain probabilities

$$g(x,y)=\begin{cases} p_{n_1}, & n_1 \\ p_{n_2}, & n_2 \\ 1-(p_{n_1}+p_{n_2}), & f(x,y) \end{cases}$$

- The corruption rate is $\left(p_{n_1}+p_{n_2}\right)\cdot 100\%$

Random Impulse Noise

- Random impulse noise replaces the intensity values in the image $f(x,y)$ by uniformly distributed random numbers with some certain probability

$$g(x,y) = \begin{cases} p_n, & \eta(x,y) = random\left[\eta_{\min}, \eta_{\max}\right] \\ 1 - p_n, & f(x,y) \end{cases}$$

- $\eta_{\min}, \eta_{\max}$ are min and max intensities of impulses
- The corruption rate is $p_n \cdot 100\%$